**COEN 178 Intro to Database Systems Winter 2019**

**Lab 7 (60 pts)**

**Objectives: Learn**

* How to fix “mutating table errors” caused by triggers.
* How to generate a formatted report using SQL commands.

**Part 1**

In this part, you will learn about **mutating table errors** caused by triggers and some solutions.

Mutating table exceptions occur when we try to reference the triggering table in a query from within row-level trigger code. In other words, a mutating table error (*ORA-04091: table name is mutating, trigger/function may not see it*) occurs when a row-level trigger tries to query or change a table that is already undergoing change (via an INSERT, UPDATE, or DELETE statement).

For example, inserting a row in a table triggers an action to calculate the summary of a column in the same table.

Since, it is the row-level triggers that cause the mutating table errors, a row-level trigger may not read or write the table from which it has been fired. However, statement level triggers are free to both read and modify the triggering table.

**Exercise 1 (10 pts)**

Step 1: Create a table as shown below:

Create table ItemOrder (orderNo VARCHAR(5) Primary key, qty Integer);

Step 2: Now define a trigger as shown below (you can copy and paste the code at QL prompt)

CREATE OR REPLACE **TRIGGER** ItemOrder\_after\_insert\_trig

AFTER INSERT

ON ItemOrder

FOR EACH ROW

DECLARE

v\_quantity Integer;

BEGIN

SELECT qty

INTO v\_quantity

FROM ItemOrder

WHERE orderNo = 'o1';

END;

/

Show Errors;

**Step 3:**

Try to insert the following row into ItemOrder.

Insert into ItemOrder values ('o1',100);

Did you succeed? What did you see?

ERROR at line 1:

ORA-04091: table ECHENG.ITEMORDER is mutating, trigger/function may not see it

ORA-06512: at "ECHENG.ITEMORDER\_AFTER\_INSERT\_TRIG", line 5

ORA-04088: error during execution of trigger

'ECHENG.ITEMORDER\_AFTER\_INSERT\_TRIG'

The solution to the above problem is not to include the query in the trigger. But sometimes, it becomes necessary to include a query statement to enforce a constraint.

**Exercise 2 (15 pts)**

Create the tables **COURSE** and **COURSE\_PREREQ** as shown below, where COURSE contains the information about a course with a courseNo and name.

The Course\_Prereq has the courseNo and its prerequisite (which is a Course).

Why do you think the prerequisites are in a separate table and not included in the Course table? (5 pts)

The prerequisites should not be included in the course table because it cannot reference itself.

**CREATE TABLE Course**

**(**

**courseNo INTEGER PRIMARY KEY,**

**courseName VARCHAR(20)**

**);**

**CREATE TABLE Course\_Prereq**

**(**

**courseNo INTEGER ,**

**prereqNo Integer,**

**Foreign Key(prereqNo) references Course (courseNo)**

**);**

insert into Course values (10,'C++');

insert into Course values (11,'Java');

insert into Course values (12,'Python');

insert into Course values (121,'Web');

insert into Course values (133,'Software Eng');

**Now, we want to enforce a constraint that *a course cannot have more than 2 prerequsites.***

**Write the following trigger to enforce the constraint.**

**CREATE OR REPLACE TRIGGER LimitTest**

**BEFORE INSERT OR UPDATE ON Course\_Prereq**

**FOR EACH ROW -- A row level trigger**

**DECLARE**

**v\_MAX\_PREREQS CONSTANT INTEGER := 2;**

**v\_CurNum INTEGER;**

**BEGIN**

**BEGIN**

**SELECT COUNT(\*) INTO v\_CurNum FROM Course\_Prereq**

**WHERE courseNo = :new.CourseNo Group by courseNo;**

**EXCEPTION**

**-- Before you enter the first row, no data is found**

**WHEN no\_data\_found THEN**

**DBMS\_OUTPUT.put\_line('not found');**

**v\_CurNum := 0;**

**END;**

**if v\_curNum > 0 THEN**

**IF v\_CurNum + 1 > v\_MAX\_PREREQS THEN**

**RAISE\_APPLICATION\_ERROR(-20000,'Only 2 prereqs for course');**

**END IF;**

**END IF;**

**END;**

**/**

**SHOW ERRORS;**

Make sure that the trigger compiles without errors.

Insert the following rows into Course\_PreReq table.

insert into Course\_Prereq values (121,11);

insert into Course\_Prereq values (121,10);

Using Select, check the data in Course\_PreReq table.

Enter the row below (trying to add a third prerequisite for course 121)

insert into Course\_Prereq values (121,12);

Did you successfully add the above row into the table?

No, ERROR at line 1:

ORA-20000: Only 2 prereqs for course

ORA-06512: at "ECHENG.LIMITTEST", line 16

ORA-04088: error during execution of trigger 'ECHENG.LIMITTEST'

Using Select, check the data in Course\_PreReq table. How many rows are there?

There are two rows.

Enter the row below.

insert into Course\_Prereq values (133,12);

Using Select, check the data in Course\_PreReq table. How many rows are there?

There are now three tables.

Now, do an update as shown below:

update COURSE\_PREREQ

set courseno = 121 where courseno= 133;

**What is the result of update above? Did it work? Did you see any mutating table error?**

**It is possible to insert (single-row inserts), but not update without causing a mutating trigger error.**

**One solution to this problem is to use Compound Triggers introduced in Oracle 11G.**

update COURSE\_PREREQ

\*

ERROR at line 1:

ORA-04091: table ECHENG.COURSE\_PREREQ is mutating, trigger/function may not see

it

ORA-06512: at "ECHENG.LIMITTEST", line 6

ORA-04088: error during execution of trigger 'ECHENG.LIMITTEST'

**Exercise 3 (10 pts)**

A compound trigger is a single trigger on a table that enables you to specify actions for each of four timing points:

1.Before the firing statement

2.Before each row that the firing statement affects

3.After each row that the firing statement affects

4.After the firing statement

Let us define a compound trigger to avoid mutating-table error.

Ref: <http://www.dbanotes.com/database-development/using-triggers-and-compound-triggers-in-oracle-11g/>

**Do the following:**

Step 1: Delete all rows from Course\_Prereq table.

Step 2: Define the following compound trigger.

CREATE OR REPLACE TRIGGER LimitTest

FOR INSERT

ON Course\_Prereq

COMPOUND TRIGGER

/\* Declaration Section\*/

v\_MAX\_PREREQS CONSTANT INTEGER := 2;

v\_CurNum INTEGER := 1;

v\_cno INTEGER;

--ROW level

BEFORE EACH ROW IS

BEGIN

v\_cno := :NEW.COURSENO;

END BEFORE EACH ROW;

--Statement level

AFTER STATEMENT IS

BEGIN

SELECT COUNT(\*) INTO v\_CurNum FROM Course\_Prereq

WHERE courseNo = v\_cno Group by courseNo;

IF v\_CurNum > v\_MAX\_PREREQS THEN

RAISE\_APPLICATION\_ERROR(-20000,'Only 2 prereqs for course');

END IF;

END AFTER STATEMENT;

END ;

/

SHOW ERRORS;

Step 3: Compile the error and fix any errors.

Step 4: Insert the following rows.

insert into Course\_Prereq values (121,11);

insert into Course\_Prereq values (121,10);

insert into Course\_Prereq values (121,12);

insert into Course\_Prereq values (133,12);

Step 5: Do a select and display the data in Course\_Prereq. Is the constraint, ***a course cannot have more than 2 prerequsites,* enforced?**

Yes it is enforced, ERROR at line 1:

ORA-20000: Only 2 prereqs for course

ORA-06512: at "ECHENG.LIMITTEST", line 21

ORA-04088: error during execution of trigger 'ECHENG.LIMITTEST'

Step 6: Do an update as shown below.

update COURSE\_PREREQ

set courseno = 121 where courseno= 133;

What is the result? Do a select and display the data in Course\_Prereq. Is the constraint, ***a course cannot have more than 2 prerequsites,* enforced?**

Yes

COURSENO PREREQNO

---------- ----------

121 11

121 10

121 12

**Part 2**

**Exercise 4 (5 pts)**

Run **report.sql** (Start report.sql). Examine the code.

**Exercise 5 (20 pts)**

Do **Lab8\_Report\_Exercise**. Follow the instructions given in the file.

For your reference, read **Creating\_A\_Report.doc**